逻辑与神经之间的桥

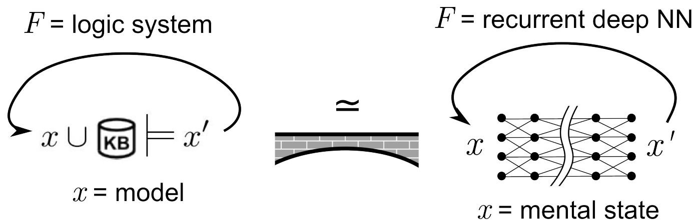
甄景贤 (King-Yin Yan)

General.Intelligence@Gmail.com

Abstract. Logic-based AI 和 connectionist AI 长久分裂，但作者最近发现可以建立两者之间的对应关系。逻辑的结构类似人类的自然语言，但大脑是用神经思考的。机器学习的主要目标，是用 inductive bias 去加快学习速度，但这目标太空泛。将逻辑结构加到神经结构之上，就增加了约束，亦即 inductive bias。

逻辑 AI 那边，「结构」很抽象符号化，但**学习算法**太慢；我的目的是建立一道「桥」，将逻辑 AI 的某部分结构**转移**到神经网络那边。

这个问题搞了很久都未能解决，因为逻辑 AI 那边的结构不是一般常见的数学结构，单是要要表述出来也有很大困难。直到我应用了 model theory 的观点，才找到满意的解决方案：



(1)

首先解释 neural network 那边的结构，然后再解释 logic 那边的结构。

1 神经网络的结构

一个 神经网络 基本上是：

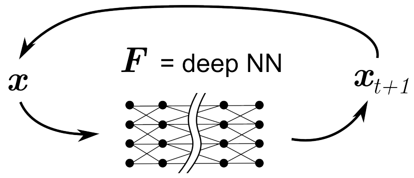
$$F(\boldsymbol{x}) = \sigma(W_1 \sigma(W_2 ... \sigma(W_L\, \boldsymbol{x}))) \tag{2}$$

其中 L 是层数，W_ℓ 是每层的**权重**矩阵，σ 是对每个分量的 sigmoid function（其作用是赋予非线性）。

σ 作用在 $\boldsymbol{x}$ 的每个分量上，它的作用在座标变换下**没有不变性**。所以 σ 不是一个向量运算，从而 $\mathbb{X} \ni \boldsymbol{x}$ 的结构也不是**向量空间**的结构。通常习惯把 $\vec{x}$ 写成向量形式，但这有点误导。

如果将神经网络**首尾相接**造成迴路，这是一种智能系统的最简单形式。举例来说，如果要识别「白猫追黑猫」的视像，「猫」这物体要被识别两次，很明显我们不应浪费两个神经网

络去做这件事，所以 迴路 是必须的：



(3)

它的状态方程是：

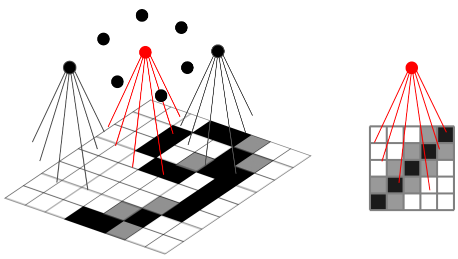
$$\boxed{\text{离散时间}} \qquad \boldsymbol{x}_{t+1} = \boldsymbol{F}(\boldsymbol{x}_t) \tag{4}$$

$$\boxed{\text{连续时间}} \qquad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{5}$$

由此可以看出，$\mathbb{X}$ 是一个**微分流形**。更深入地讲，它是一个力学上的 Hamiltonian 系统，具有 symplectic（辛流形）结构。但这超出了本文範围，详见本篇的姊妹作 [13]。

现在思考一下，神经网络怎样识别模式，或许会有帮助....

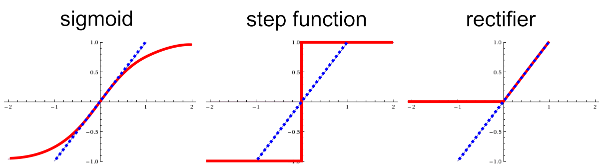
考虑最简单的情况，例如提取数字 “9” 的特徵的一层网络。这层网络可以有很多神经元（左图），每个神经元局部地覆盖输入层，即所谓视觉神经元的 local receptive field（右图）。



(6)

假设红色的神经元专门负责辨识「对角线」这一特徵。它的方程式是 $\boldsymbol{y} = \sigma(W\boldsymbol{x})$。矩阵 W 的作用是 affine「旋转」特徵空间，令我们想要的特徵指向某一方向。然后再用 σ 「**挤压**」想要的特徵和不想要的特徵。Sigmoid 之后的输出，代表某类特徵的**存在与否**，即 $\{0, 1\}$。这是一种资讯的**压缩**。

讲一点 chaos theory：σ^{-1} 的作用是「扯」(stretch)，将本来邻近的两点的距离非线性地拉远。看看以下各种常见的激活函数，它们全都是相对於 identity $y = x$ 的非线性 deformation：



(7)

这和 Steven Smale 提出的「马蹄」[10] 非常类似，它是制造混沌的处方之一。Smale 马蹄的另一个变种叫做 baker map，其作用类似於「搓面粉」。换句话说，「拉扯」然后放回原空间，如此不断重复，就会产生混沌 [5] [11]。（神经网络的时间逆向就是 σ^{-1}，所以时间向前也是混沌。）

这里有一点重大意义：$\boldsymbol{F}^{-1}$ 有混沌的典型特徵：它「对初始状态的微少变化非常敏感」。换句话说 $\boldsymbol{F}$ 的逆是 unpredictable 的，简言之，神经网络 $\boldsymbol{F}$ 是**不可逆**的压缩过程。

最近一个有趣的例子是 DeepDream [1]，它用神经网络的 $\boldsymbol{F}^{-1}$ 产生有迷幻感觉的 pre-image，证实了 $\boldsymbol{F}^{-1}$ 可以和原本的 image 差距很大：

(8)

总括以上，可以归结出一些原则：

• <u>每个神经元的输出代表某个 feature 的存在与否</u>
• <u>更高层的神经元代表下层 features 之间的**关系**</u> (9)

这些原则暂时未能严格地表述和证明，或者可以叫它做 **神经原理 (Neural Postulate)**。

凭这个思路推广，可以推测这样的 correspondence：

$$\begin{array}{rcl} \mathbb{M} & \simeq & \mathbb{X} \\ \text{逻辑物体} & \Leftrightarrow & \text{neuron} \\ \text{逻辑关系} & \Leftrightarrow & \text{relation between higher and lower neurons} \end{array} \tag{10}$$

2 逻辑的结构

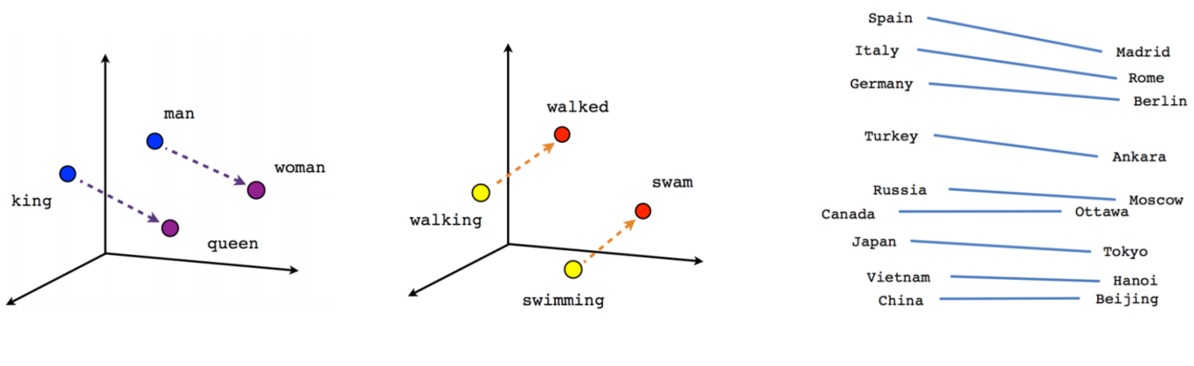
一个**逻辑系统**可以这样定义：

- 一些 constant symbols，predicate symbols，和 function symbols
- 由上述的原子建立 **命题** (propositions)
- 命题之间可以有连接词：$\neg, \wedge, \vee$ 等
- 建立 **逻辑后果** (consequence) 关系：$\Gamma \vdash \Delta$

2.1 Semantic distance 与 Compositionality

我们的目的是想将整套逻辑 AI 的器材搬到 连续 的微分流形上去实现。这个做法，是受了 Google 的 Word2Vec [8] 算法的启发，它可以将自然语言的词语 embed 到度量空间中，而

且词语之间的距离是 semantic distance （语义学上的距离）：



Male-Female　　Verb tense　　Country-Capital　　(11)

当然，word2vec 一经发表之后，很多人开始构思怎样把「句子」也 embed 到度量空间上。第一个想法当然是用 tensor product，例如 “he loves her” 这句子就变成了 $\text{he} \otimes \text{loves} \otimes \text{her}$。

但这个做法很有问题；在语言学 / 逻辑学里有一个重要概念叫 compositionality，它说：

$$\underline{\text{合成物的语义 (semantics) 由其所构成的原子生成，而不依赖於其他资料}} \tag{12}$$

例如「我爱你」由「我、爱、你」三个原子组成。如果原子用向量表示，那么「我爱你」和「没有你我不能生存」这两句意义相近，但形式上 (syntax) 却很不同，导致这两组原子的 tensor products 可能相距颇远。

用 tensor product 的做法，得出的空间结构是 syntactic 而不是 semantic 的，但神经网络学习的重点是 泛化 (generalization)，它基於「邻近的点的意义相近」才能成功，所以一定要 semantic distance。

2.2 「二分法」逻辑

上节看到 tensor product 的做法不行，它用原子 “algebraically” 组合成句子，原子之间的距离可以是语义学相近的，但「乘出来」的句子未必有 semantic distance。我们想要的是相反的特性：句子之间要有 semantic distance，但这样似乎不能将句子分拆成原子，尤其是日常语言中的词语 (words)。

暂时只考虑一个 thought （= 命题）如何表示。

假设思维空间的 metric 是语义的（语义相近则点的距离相近）。根据「神经原理」(9)，每个 thought 必然是由一组 features 表述，例如 $\boldsymbol{x} = (x_1, x_2, ..., x_n)$，这里每个 x_i 就好比一个概念原子。但问题是我们不想 thought 用原子「乘出来」，怎办？

记得逻辑学历史中 George Boole 曾经提出过一种逻辑 [1]，例如 x 代表「是人的」，y 代表「会死的」，那么「所有人都会死」就是：

$$x\,(1-y) = 0 \tag{13}$$

[1] 这和后世为他命名的 Boolean logic 不同。

这里 x, y 是 “classes”，乘法是集合的 intersection，即 $\cap$。这种乘法和上面描述的句子 / 词语乘法不同；首先，这乘法是 commutative 的，$x \cap y = y \cap x$。如果将一个 thought 表示为：

$$\boldsymbol{x} = x_1, x_2, ..., x_n = x_1 \cap x_2 ... \cap x_n \tag{14}$$

则这些 x_i 可以看成是将所有思维的空间用「二分法」切割，每个 x_i 是一个二分切割。

两个 thoughts 之间的距离可以用 Hamming distance 量度，它是一个 semantic distance，尤其是当 n 比较大时，可能是一个效果不错的 metric。

总括来说：「二分法」逻辑用一组 binary features 去定义一个 thought，一条思维是 n-维 hypercube 上的一个顶点。

2.3 逻辑中的 linkage 现象

在经典逻辑中有一个”linkage” 的现象，例如以下这个式子：

$$\text{grandfather}(X,Z) \longleftarrow \text{father}(X,Y) \wedge \text{father}(Y,Z) \tag{15}$$

那意思是说，无论右边的变量（例如 X）怎样改变，左边的 X 必须代入相同的值。这是 代入 (substitution) 的本质。

简单来说，需要用一些 functions 描述这些逻辑式子。当然，可以用完全无限制 (free) 的函数，但我不想丧失结构（因为这些函数的整个家族都有此 linkage 结构，每次重复学习同一种结构是浪费时间的）。

虽然 (15) 是 first-order predicate logic，我们可以用类似的方法泡制「二分法」逻辑中的 linkage，其功效或许差不多吧？

换句话说，命题就是 thought，一个 conditional formula 是一个函数，它将几个（前提）命题映射到一个（结论）命题。在这函数里面有 linkage 结构，这 linkage 结构是 positional 的，亦即是说，一个 linkage 是由 source 空间的第 p 命题的第 q 个分量到 target 空间的第 r 个分量的 projection function π。

一个没有 linkage 的函数 f 可以「投影」到某个 projection function 变成有 linkage：

$$f_r \mapsto \pi_{p,q} \tag{16}$$

但这只是一条 rule，rules 构成 KB，但似乎 $\boldsymbol{F}$ 应该包含整个 KB。

2.4 用 relation algebra 如何？

以前曾经对 relation algebra [9] [7] 有些憧憬，因为它比较接近人类自然语言，但其实 relation algebra (RA) 和 first-order logic (FOL) 基本上是等效的，在 FOL 里面有 linkage 的复杂性，但在 RA 里面这个复杂性其实也没有消失。可以说「复杂度是守恒的」。

举例来说，在 (15) 中表达「爸爸的爸爸是爷爷」，可以用 RA 更简单地表达：

$$\text{father} \circ \text{father} = \text{grandfather} \tag{17}$$

实际的推导是这样的：

$$\text{John father Pete} \tag{18}$$
$$\text{Pete father Paul} \tag{19}$$
$$\text{John father} \circ \text{father Paul} \tag{20}$$

这时要**代入**上面的等式才能得出结论

$$\text{John grandfather Paul} \tag{21}$$

所以 linkage 的复杂性变成了代数 formula 长度的复杂性。

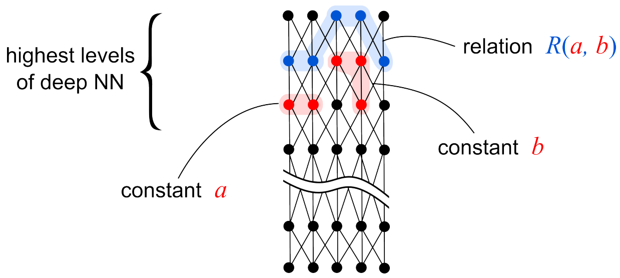
2.5 命题的集合

神经的状态空间由一些 **thoughts**（思维）组成，例如「我正在上课 $\wedge$ 邻座的女孩很漂亮」，这两个 thoughts 是独立的。也可以有另一个状态：「我正在搭地铁 $\wedge$ 邻座的女孩很漂亮」。Thoughts 独立的好处是表述的 economy。思维状态 $\boldsymbol{x}$ 就是 M 个 thoughts 的集合，M 是 working memory 的大小。认知科学里有个说法：the size of human working memory is approximately 7 ± 2 items.

思维空间是 M 个 hypercube 的卡积。

3 结论

但要注意的是这对应未必是一对一的，可能是一个 constant 对应几个 neurons 的（线性？）**组合**。具体情况可能像以下的示意图（实际上每层神经网络可能有很多神经元）：



(22)

$R(a, b)$ 可以在 a, b 的 common parents 中寻找（例如那些蓝色神经元，$R(a, b)$ 的值 = 蓝色神经元的某个线性组合）。验证的方法是：当 a 和 b 的信号都是「有」时，$R(a, b)$ 的值也应该是 true。

这篇论文并不太成功，因为跳到 (10) 和 (18) 的结论没有严谨的根据，只是直观上觉得有可能。理论上来说，既然知道了 $\mathbb{M}$ 那边是怎样生成的、$\mathbb{X}$ 那边是怎样生成的，则要在两边建立「高速公路」应该是可行的。实际上，似乎只要建立一个深度网络就可以，因为神经网路是 universal function approximator，根本不用考虑 $\mathbb{M}$ 和 $\mathbb{X}$ 这两个结构之间的关系。

进一步的研究，希望数学专业的人能帮助一下：

1. 在逻辑那边，可不可以转换成 algebraic geometry 的结构？即是说：逻辑式子 $\simeq$ 代数方程。这种代数逻辑的做法，我暂时只知道有 [3]，是很偏门的研究。

2. 能不能根据 $\mathbb{M}$ 和 $\mathbb{X}$ 的结构，找出它们之间的桥的最简单形式？可以用数学归纳法，逐步考虑 $\mathbb{M}$ 和 $\mathbb{X}$ 生成的方式，或许有帮助？

应用：对於用深度学习做 natural language understanding 的人，这理论或许会很有用。

4 Prior art

- Bader, Hitzler, Hölldobler and Witzel 在 2007 年提出了一个 neural-symbolic integration 的做法 [4]。他们首先由 logic theory 生成抽象的 Herbrand model[2]，再将 Herbrand model 映射到某个 fractal 空间，然后直接用神经网络学习那 fractal 空间。虽然用了 model theory，但他们没有利用到本文所说的 $\mathbb{M}$ 和 $\mathbb{X}$ 之间的关系。

- Khrennikov 在 1997 年开始的多篇论文中提出了用 p-adic 代数来模拟思维空间 $\mathbb{X}$ 的结构，详见 [2] 一书。一个 p-adic 数可以看成是一个 p 进制的小数，p 是任何质数。

- 经典逻辑是二元逻辑，近代已经有无数将它扩充到 fuzzy 或 probablistic 的尝试（作者也提出过 [14]），但仍未有统一的理论。与此不同的另一个方向，如果将点看成是 first-order objects，谓词是点空间上的函数，直接得到 metric structures 上的连续逻辑 (continuous first-order logic) [12]，这可以看成是一种 $\mathbb{M}$ 的结构。

- 模型论中有（超滤子）ultra-filter 和 ultra-product 这些建构，它们起源於泛函分析，最近有很多横跨模型论和 Banach 空间的新研究 [6]。简单地说 ultra-product 用来将一些 models 构造出新的乘积 models。但我粗略地看过一下之后发现 ultra-product 通常涉及无穷集合，而且是很大的物体，在计算机上应用似乎不太实际。

Acknowledgement

谢谢 Ben Goertzel（OpenCog 人工智能的创始人）在 AGI mailing list 上和我的讨论。Ben 初次指出神经网络学习和逻辑 inductive 学习的不同，引起我研究两者之间的关系。

References

1. Wikipedia entry: Deep dreaming (https://en.wikipedia.org/wiki/Deepdreaming).

2. Vladimir Anashin and Andrei Khrennikov. *Applied algebraic dynamics*. de Gruyter, 2009.

3. Andreka, Nemeti, and Sain. *Handbook of philosophical logic*, chapter Algebraic logic, pages 133–247. Springer, 2001.

[2] Herbrand model 是邏輯 AI 中常用的概念，大意是用邏輯語言 $\mathcal{L}$ 生成「所有可以代入的東西」(instantiating whatever that can be instantiated)，由此產生的不含變量的句子 (sentence) 的集合。換句話說，Herbrand model 的特點是它只靠 $\mathcal{L}$ 自身產生它的模型，而不依賴任何外在結構。每个邏輯 theory 都必然至少有一个 Herbrand model。

4. Bader, Hitzler, Hödobler, and Witzel. The core method: Connectionist model generation for first-order logic programs. *Studies in Computational Intelligence 77, 205-232*, 2007.
5. Robert Gilmore and Marc Lefranc. *The topology of chaos: Alice in stretch and squeezeland*. Wiley-VCH, 2011.
6. José Iovino. *Applications of model theory to functional analysis*. Dover, 2002.
7. Roger Maddux. *Relation algebras*. Elsevier, 2006.
8. Mikolov, Sutskever, Chen, Corrado, and Dean. Efficient estimation of word representations in vector space. *Proceedings of workshop at ICLR*, 2013.
9. Gunther Schmidt. *Relational mathematics*. Cambridge, 2010.
10. Stephen Smale. Differentiable dynamical systems. *Bulletin of the American Mathematical Society*, 1967.
11. Tamás Tél and Márton Gruiz. *Chaotic dynamics: an Introduction based on classical mechanics*. Cambridge, 2006.
12. Itaï Ben Yaacov, Alexander Berenstein, C Ward Henson, and Alexander Usvyatsov. Model theory for metric structures. In *Model theory with applications to algebra and analysis, vol 2*. Cambridge, 2008.
13. King Yin Yan. Wandering in the labyrinth of thinking – a cognitive architecture combining reinforcement learning and deep learning. to be submitted AGI 2017.
14. King-Yin Yan. Fuzzy-probabilistic logic for common sense reasoning. *Artificial general intelligence 5th international conference, LNCS 7716*, 2012.